Nombres de clases en CSS

Yuri Padilha
02/12/2020

Hola, si no has visto la primera publicación de la serie, haz clic aquí y léelo antes.

Bueno, ahora solucionemos el problema de **nombres de clases CSS**, recapitulemos el componente .box que creé:

```
-- Box
.box {}
.image {}
.title {}
```

Imagine ahora que nuestro componente de la barra de navegación también tiene su propio elemento de título, ¿qué nombre le daríamos a la clase de ese título?

```
-- Navbar
.navbar {}
.title {}
```

El problema con el uso de nombres tan genéricos es que los nombres de clases como este entrarán en conflicto entre sí, asegúrese de que tanto el componente `box` cuánto `navbar` tiene un `.title` declarado. ¿Cómo podemos hacer para solucionar esto?

Piensa conmigo, ¿el título de la navbar pertenece a qué componente? ¡La navbar en sí!

¿A qué componente pertenece el título del box? ¡El mismo box!

```
-- Box
.box {}
.box .title {}
.box .image {}

-- Navbar
.navbar {}
.navbar .title {}
.navbar .item {}
```

¡Ahora sí! ¡Los estilos serán específicos para sus elementos!

Aunque hemos mejorado el código ahora, este nuevo código seguirá entristeciendo a mucha gente, especialmente a los que le harán el mantenimiento en el futuro.

Mira, pensemos en esto, ¿recuerdas nuestro elemento activo de la navbar? Entonces, pensemos que el CSS de nuestro modificador activo tiene que ser escrito por otra persona nueva en el proyecto y que ya no somos parte de ese proyecto. La persona probablemente escribirá algo como:

```
.active {}
```

Cuando inserte este modificador en el elemento de la navbar, ¿qué sucederá?

```
<navbar …>
    <a .. class="item active">
...'
```

No pasa NADA.

AH, ¿cómo es que nada? Bueno, en CSS hay algo llamado especificidad, cuanto más específico sea el CSS, más prioridad tendrá para cambiar el estilo del elemento. Mira esto:

```
<navbar class="navbar">
    <a href="#" class="item">Home</a>
    <a href="#" class="item active">Courses</a>
```

```
    <a href="#" class="item">Carreras</a>
</navbar>
```

¿Qué selectores de CSS tenemos que están estilizando a nuestro enlace courses?

```
.navbar .item {}
.active {}
```

¿Cuál de estos selectores crees que es más específico y tendrá más prioridad?

```
.navbar .item {}
```

Bueno, un selector con dos clases es más específico ¡que un selector con solo una!

Lo que sucederá es que la persona que creó el estilo .active tendrá que dejar este selector igual o incluso más específico que el selector de arriba. Resultando en:

```
.navbar .active {}
```

¿Qué pasa si después de eso alguien más quiere darle estilo? Tendrá que ser cada vez más específico hasta llegar a

.. 1 minuto de silencio ..

!important

Esto es horrible porque **!import** es un comodín para eludir la especificidad, es decir, si alguien quiere darle más estilo al elemento, ¡tendrá problemas!

Si un día te metes con temas listos para Wordpress e intentas personalizarlos, sentirás todos estos problemas en tu piel, ¡créeme!

Bueno, está bien, pero ¿cómo solucionar este problema? ¡Simple, haciendo las clases menos específicas! Intentemos dejar solo el NOMBRE del selector más específico:

```
.navbar {}
.navbar-title {}
.navbar-active {}
```

¡Observe que ahora solo tenemos una clase con un nombre compuesto! ¡Ahora .navbar-active anula .navbar-title!

Ya existe un estándar para esto y es el famoso Block Element Modifier, BEM.

Explicando las siglas de BEM tenemos:

(B)lock- Eso sería básicamente lo que **SMACSS** llama de Módulo y lo que aquí también llamamos de un componente, sería cualquier componente que creamos que se puede reutilizar en varias páginas, como nuestra box o de la navbar.

A partir de ahora, cuando me refiera a un block o bloque, estaré hablando de algún componente. **(E)lement** - Este sería un elemento que está dentro de nuestro bloque, por ejemplo nuestros elementos dentro de navbar, así como el title y la image dentro del box, ¿recuerdas?

```
-- navbar block
<navbar class="navbar">
    <a href="#" class="navbar-item">Home</a>
    <a href="#" class="navbar-item">Courses</a>
    <a href="#" class="navbar-item">Carreras</a>
</navbar>

.navbar {}
.navbar-item {}

-- box block
   <a class="box" href="#">
       <img class="box-image" src="#">
       <h3 class="box-title">Curso de HTML CSS >
   </a>

.box {}
.box-image {}
.box-title {}
```

(M) odifier - Ese sería nuestro modificador, ¿recuerdas active? navbar-itemActive.

Lo que propone BEM es organizar estos nombres, ¿qué nombres le vamos a dar a los elementos? ¿Y los modifiers? ¿Y a los blocks? Ayuda a las personas a seguir un estándar. Para BEM, este estándar sería:

```
block__element—modifier
```

Es decir, refactorizando nuestro código tendríamos: Para el block box:

```
.box {}
.box__title {}
.box__image {}
```

Vea que en el ejemplo del box no tenemos un modifier, ¡solo el block box y los *elements* `title` e `image`!

Vayamos a nuestro block navbar ahora:

```
.navbar {}
.navbar__item {}
.navbar__item--active {}
```

¿Recuerdas que active es nuestro modifier? ¡Voilà!

¡Hicimos el BEM en nuestro CSS! ¡Así es como lo hacemos en Alura hoy, ¡utilizando tanto las técnicas descritas en esta publicación como las técnicas descritas en la publicación anterior sobre la organización de archivos!

En estas dos publicaciones solo hablamos de HTML y CSS, para javascript ya es de otra manera, ¡quizás también escriba una publicación hablando un poco sobre JS en el futuro! Solo lo complementaré con algunas últimas informaciones:

BEM no es restrictivo, es decir, si separas sus bloques de elements y de modifiers, ¡ya está siguiendo un estándar BEM! Mira, en Alura solíamos seguir el estándar:

```
Block-element--modifier
```

Este estándar funcionó durante un tiempo, pero a medida que el código crecía, comenzamos a encontrarlo confuso, la diferencia entre element y modifier es solo 1 guión.

Esto empezó a ser un problema. ¡Si haces un inspect en algunas páginas de Alura hoy, todavía encontrarás un código como ese!

¡Gracias por leer y un gran abrazo!

ARTÍCULOS DE TECNOLOGÍA > FRONT END

CURSOS

Cursos de Programación
Lógica de Programación | Java

Cursos de Front End
HTML y CSS | JavaScript | React

Cursos de Data Science
Data Science | Machine Learning | Excel | Base de Datos | Data Visualization | Estadística

Cursos de DevOps
Docker | Linux

Cursos de Innovación y Gestión
Productividad y Calidad de Vida | Transformación Ágil | Marketing Analytics | Liderazgo y Gestión de Equipos | Startups y Emprendimiento